Winner Best Paper, International Parallel & Distributed Processing Symposium (IPDPS), March 24-30, 2007, Long Beach, CA.

Scientific Application Performance on Candidate PetaScale Platforms

Leonid Oliker[1], Andrew Canning[1], Jonathan Carter[1], Costin Iancu[1], Michael Lijewski[1], Shoaib Kamil[1], John Shalf[1], Hongzhang Shan[1], Erich Strohmaier[1], Stéphane Ethier[2], Tom Goodale[3]

[1]Computational Research Division / NERSC
Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA

[2]Princeton Plasma Physics Laboratory
Princeton University
Princeton, NJ 08453, USA

[3]Computer Science, Cardiff University
The Parade, CF24 4QJ, UK &
CCT, LSU, LA 70803, USA

Abstract

After a decade where HEC (high-end computing) capability was dominated by the rapid pace of improvements to CPU clock frequency, the performance of next-generation supercomputers is increasingly differentiated by varying interconnect designs and levels of integration. Understanding the tradeoffs of these system designs, in the context of high-end numerical simulations, is a key step towards making effective petascale computing a reality. This work represents one of the most comprehensive performance evaluation studies to date on modern HEC systems, including the IBM Power5, AMD Opteron, IBM BG/L, and Cray X1E. A novel aspect of our study is the emphasis on full applications, with real input data at the scale desired by computational scientists in their unique domain. We examine six candidate ultra-scale applications, representing a broad range of algorithms and computational structures. Our work includes the highest concurrency experiments to date on five of our six applications, including 32K processor scalability for two of our codes and describe several successful optimizations strategies on BG/L, as well as improved X1E vectorization. Overall results indicate that our evaluated codes have the potential to effectively utilize petascale resources; however, several applications will require reengineering to incorporate the additional levels of parallelism necessary to achieve the vast concurrency of upcoming ultra-scale systems.

1 Introduction

Computational science is at the dawn of petascale computing capability, with the potential to achieve simulation scale and numerical fidelity at hitherto unattainable levels. However, harnessing such extreme computing power will require an unprecedented degree of parallelism both within the scientific applications and at all levels of the underlying architectural platforms. Unlike a decade ago — when the trend of HEC (high-end computing) systems was clearly towards building clusters of commodity components — today one sees a much more diverse set of HEC models. Increasing concerns over power efficiency is likely to further accelerate recent trends towards architectural diversity through new interest in customization and tighter system integration. Understanding the tradeoffs of these computing paradigms, in the context of high-end numerical simulations, is a key step towards making effective petascale computing a reality. The main contribution of this work is to quantify these tradeoffs by examining the effectiveness of various architectural models for HEC with respect to absolute performance and scalability across a broad range of key scientific domains.

A novel aspect of our effort is the emphasis on full applications, with real input data at the scale desired by computational scientists in their unique domain, which complements a number of other related studies [4, 9, 21]. Our application suite includes a broad spectrum of numerical methods and data-structure representations in the areas of Magnetic Fusion (GTC), Fluid Dynamics (ELBM3D), Astrophysics (Cactus), High Energy Physics (BeamBeam3D), Materials Science (PARATEC), and AMR Gas Dynamics (HyperCLaw). We evaluate performance on a wide range of architectures with varying degrees of component customization, integration, and power consumption, including: the Cray X1E customized parallel vector-processor, which utilizes a tightly-coupled custom interconnect; the commodity IBM Power5 and AMD dual-core Opteron processors integrated with custom fat-tree based Federation and 3D-torus based XT3 interconnects, respectively; the com-

Name	Local	Arch	Network	Network Topology	Total P	P/ Node	Clock (GHz)	Peak (GF/s/P)	Stream BW (GB/s/P)	Stream (B/F)‡	MPI Lat (μsec)	MPI BW (GB/s/P)
Bassi	LBNL	Power5	Federation	Fattree	888	8	1.9	7.6	6.8	0.85	4.7	0.69
Jaguar	ORNL	Opteron	XT3	3DTorus	10,404	2§	2.6	5.2	2.5	0.48	5.5*	1.2
Jacquard	LBNL	Opteron	InfiniBand	Fattree	640	2	2.2	4.4	2.3	0.51	5.2	0.73
BG/L	ANL	PPC440	Custom	3DTorus	2,048	2	0.7	2.8	0.9	0.31	2.2†	0.16
BGW	TJW	PPC440	Custom	3DTorus	40,960	2	0.7	2.8	0.9	0.31	2.2†	0.16
Phoenix	ORNL	X1E	Custom	Hcube	768	8¶	1.1	18.0	9.7	0.54	5.0	2.9

Table 1. Architectural highlights of studied HEC platforms.

Name	Lines	Discipline	Methods	Structure
GTC	5,000	Magnetic Fusion	Particle in Cell, Vlasov-Poisson	Particle/Grid
ELBD	3,000	Fluid Dynamics	Lattice Boltzmann, Navier-Stokes	Grid/Lattice
CACTUS	84,000	Astrophysics	Einstein Theory of GR, ADM-BSSN	Grid
BeamBeam3D	28,000	High Energy Physics	Particle in Cell, FFT	Particle/Grid
PARATEC	50,000	Material Science	Density Functional Theory, FFT	Fourier/Grid
HyperCLaw	69,000	Gas Dynamics	Hyperbolic, High-order Godunov	Grid AMR

Table 2. Overview of scientific applications examined in our study.

modity Opteron processor integrated with the InfiniBand high-performance commodity network; and the IBM Blue Gene/L (BG/L) which utilizes a customized SOC (system on chip) on commodity, low-power embedded cores, combined with multiple network interconnects.

This work represents one of the most comprehensive performance evaluation studies to date on modern HEC platforms. For five of our six studied applications, we present the highest concurrency results ever conducted, and show that the BG/L can attain impressive scalability characteristics all the way up to 32K processors on two of our applications. We also examine several application optimizations, including BG/L processor and interconnect mappings for the SciDAC [17] GTC code, which achieve significant performance improvements over the original superscalar version. Additionally, we implement several optimizations for the HyperCLaw AMR calculation, and show significantly improved performance and scalability on the X1E vector platform, compared with previously published studies. Overall, we believe that these comprehensive evaluation efforts lead to more efficient use of community resources in both current installations and in future designs.

2 Target Architectures and Scientific Applications

Our evaluation testbed consists of six production HEC systems, including: Bassi, the Lawrence Berkeley National Laboratory (LBNL) IBM Power5-based system interconnected via IBM's HPS Federation, containing 888 compute processors (111 8-way nodes) and running AIX 5.2; Jaguar, the Oak Ridge National Laboratory (ORNL) dual-core AMD Opteron XT3 systems, containing 10,400 processors (5,200 2-way nodes) and running Catamount 1.4.22; Jacquard, the LBNL (single-core) Opteron-based system, interconnect via Infiniband with 640 processors (320 2-way nodes) and running Linux 2.6.5; BG/L, the Argonne National Laboratory (ANL) IBM PowerPC 440-based system, containing 2,048 processors (1024 2-way nodes) interconnected via three independent networks and running SuSE Linux OS (SLES9); BGW a large (40K processor) BG/L installation located at IBM's Thomas J. Watson (TJW); and Phoenix, the ORNL vector-based X1E platform, containing 768 processors (96 8-way MSP nodes) interconnected via the Cray custom switch and running UNICOS/mp 3.0.23.

Table 1 presents several key architectural features of our evaluated test suite, including: STREAM benchmark results [18] showing the measured EP-STREAM [11] triad bandwidth when all processors within a node simultaneously compete for main memory; the ratio of STREAM bandwidth to the peak computational rate; the measured inter-node MPI latency [5]; and the measured bidirectional MPI bandwidth per processor pair when each processor simultaneously exchanges data with a distinct processor in another node. Note that our BG/L measurements primarily examine performance in *coprocessor mode* where one

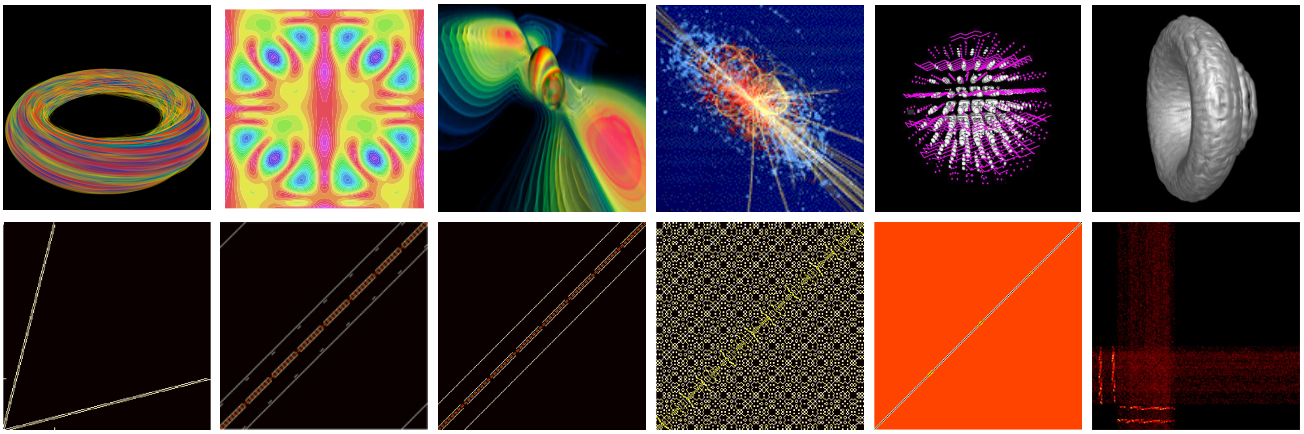
‡Ratio of STREAM bandwidth to peak processor computational rate.

§Each Jaguar node consists of a single, dual-core processor.

¶An MSP is defined as a processor for the X1E data.

*Minimum latency for the XT3 torus. There is a nominal additional latency of 50ns per hop through the torus.

†Minimum latency for the BG/L torus. There is an additional latency of up to 69ns per hop through the torus.



(a) (b) (c) (d) (e) (f)

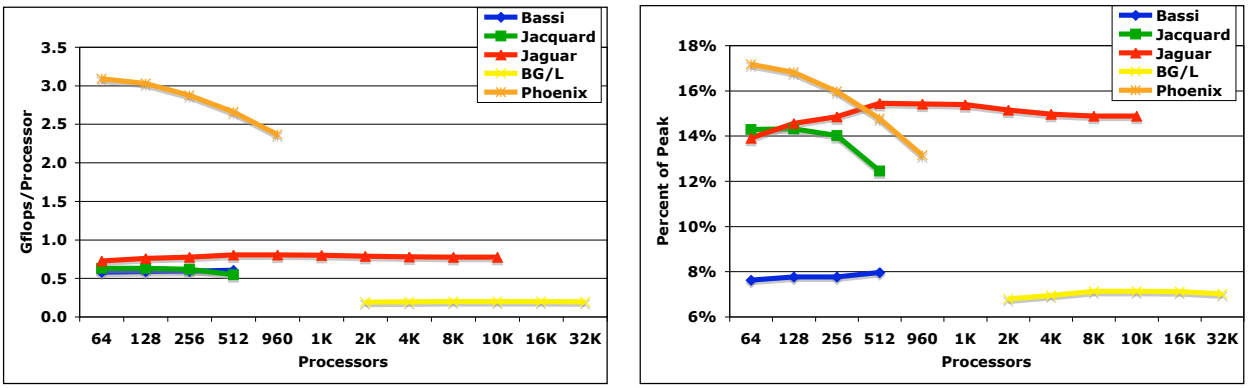
Figure 1. TOP: Visualization of (a) GTC electrostatic potential field (b) ELBM3D vorticity turbulence (c) Cactus black hole collisions (d) BeamBeam3D particle tracks (e) PARATEC CdSe quantum dot electron state and (f) HyperCLaw Helium bubble deformation. BOTTOM: Interprocessor communication topology and color-coded intensity of corresponding application.

core is used for computation and the second is dedicated to communication. Additionally, several experiments were conducted using up to 32K processors on IBM's Thomas J. Watson (TJW) BGW in *virtual node mode* where both cores are used for both computation and communication.

Six applications from diverse areas in scientific computing were chosen to compare the performance of our suite of leading supercomputing platforms. We examine: GTC, a magnetic fusion application that uses the particle-in-cell approach to solve non-linear gyrophase-averaged Vlasov-Poisson equations; ELBM3D, a lattice-Boltzmann code to study turbulent fluid flow; Cactus, an astrophysics framework for high-performance computing that evolves Einstein's equations from the Theory of General Relativity; BeamBeam3D, a parallel particle-field decomposition based particle-in-cell code for high energy ring colliders; PARATEC, a first principles materials science code that solves the Kohn-Sham equations of density functional theory to obtain electronic wave functions; HyperCLaw, an adaptive mesh refinement (AMR) framework for solving the Hyperbolic conservation laws of gas dynamics via a higher-order Godunov method. Table 2 presents an overview of the application characteristics and Figure 1 (top) shows a variety visualizations from our evaluated simulations.

These codes are candidate ultra-scale applications with the potential to fully utilize leadership-class computing systems, and represent a broad range of algorithms and computational structures. Figure 1 (bottom) presents the topological connectivity of communication for each code — where each point in the graph indicates message exchange and (color coded) intensity between two given processors — highlighting the vast range of communication requirements within our application suite. Communication characteristics include: nearest-neighbor and allreduce communication across the toroidal grid and poloidal grid (respectively) for the particle-in-cell GTC calculation; simple ghost boundary exchanges for the stencil-based ELBM3D and Cactus computations; global gather and broadcast operations to compute the charge and field properties in BeamBeam3D; all-to-all data transpositions used to implement PARATEC's 3D FFTs, and complex data movements required to create and dynamically adapt grid hierarchies in HyperCLaw. Examining these varied computational methodologies across a set of modern supercomputing platforms allows us to study the performance tradeoffs of different architectural balances and topological interconnect approaches.

Experimental results show either strong scaling (where the problem size remains fixed regardless of concurrency), or weak scaling (where the problem size grows with concurrency such that the per-processor computational requirement remains fixed) — whichever is appropriate for a given application's large-scale simulation. Note these applications have been designed and highly optimized on superscalar platforms; thus, we describe newly devised optimizations for the vector platforms where appropriate. Performance results measured on these systems, presented in Gflop/s per processor (denoted as Gflops/P) and percentage of peak, are used to compare the time to solution of our evaluated platforms. The Gflop/s value is computed by dividing a valid baseline flop-count by the measured wall-clock time of each platform — thus the ratio between the computational rates is the same as the ratio of runtimes across the evaluated systems. All results are shown using the fastest (optimized) available code versions.



(a) (b)

Figure 2. GTC weak-scaling performance using 100 particles per cell per processor (10 for BG/L) in (a) Gflops/processor and (b) percentage of peak. All BG/L data collected on the BGW system.

3 GTC: Particle-in-Cell Magnetic Fusion

GTC is a 3D particle-in-cell code developed for studying turbulent transport in magnetic confinement fusion plasmas [6, 12]. The simulation geometry is that of a torus, which is the natural configuration of all tokamak fusion devices. As the charged particles forming the plasma move within the externally-imposed magnetic field, they collectively create their own self-consistent electrostatic (and electromagnetic) field that quickly becomes turbulent under driving temperature and density gradients. The particle-in-cell (PIC) method describes this complex interaction between fields and particles by solving the 5D gyro-averaged kinetic equation coupled to the Poisson equation. In the PIC method, the interaction between particles is calculated using a grid on which the charge of each particle is deposited and then used in the Poisson equation to evaluate the field. This is the scatter phase of the PIC algorithm. Next, the force on each particle is gathered from the grid-base field and evaluated at the particle location for use in the time advance.

GTC utilizes two parallel algorithms. The first one, originally implemented in GTC, is a one-dimensional domain decomposition in the toroidal direction (long way around the torus) while the second is a particle distribution within each domain. The processors in charge of the same domain have a copy of the local grid, essentially a single plane, but hold a fraction of the particles and are linked with each other by a domain-specific communicator used when updating grid quantities calculated by individual processors. In the toroidal direction, a second communicator links a single processor in each domain in a ring-like fashion. The visual representation of the communication topology can been seen in Figure 1(a).

3.1 Experimental results

Figure 2 shows the results of a weak-scaling study of GTC on the platforms under comparison in both (a) raw performance and (b) percentage of peak. The size of the grid remains fixed since it is prescribed by the size of the fusion device being simulated, while the number of particles is increased in a way that keeps the same amount of work per processor for all cases.

Looking at the raw performance we see that the Phoenix platform clearly stands out with a Gflops/P rate up to 4.5 times higher than the second highest performer, the XT3 Jaguar. This was expected since the version of GTC used on Phoenix has been extensively optimized to take advantage of the multi-streaming vector processor [13]. In the latest improvements, the dimensions of the main arrays in the code have been reversed in order to speed up access to the memory banks, lead to higher performance. This change is not implemented in the superscalar version since it reduces cache reuse and hence slows down the code. Although still high, the performance per processor on the X1E decreases significantly as the number of processors, or MSPs, increases. This is probably due to the increase in intra-domain communications that arises when the number of processors per toroidal domain increases. An *Allreduce* operation is required within each domain to sum up the contribution of each processor, which can lead to lower performance in certain cases. Optimizing the processor mapping is one way of improving the communications but we have not explored this avenue on Phoenix yet.

Jacquard, Bassi, and Jaguar have very similar performance in terms of Gflops/P although Bassi is shown to deliver only about half the percentage of peak achieved on Jaguar, which displays outstanding efficiency and scaling all the way to 5184 processors. The percentage of peak achieved by particle-in-cell codes is generally low since

the gather-scatter algorithm that characterizes this method involves a large number of random accesses to memory, making the code sensitive to memory access latency. However, the AMD Opteron processor used in both Jacquard and Jaguar delivers a significantly higher percentage of peak for GTC compared to all the other superscalar processors. It even rivals the percentage of peak achieved on the vector processor of the X1E Phoenix. This higher GTC efficiency on the Opteron is due, in part, to relatively low main memory latency access. On all systems other than Phoenix, GTC exhibits near perfect scaling, including up to 5K processors on Jaguar.

The percentage of peak achieved by GTC on BG/L is the lowest of the systems under study but the scalability is very impressive, all the way to 32,768 processors! The porting of GTC to the BG/L system was straightforward but initial performance was disappointing. Several optimizations were then applied to the code, most of them having to do with using BG/L-optimized libraries such as MASS and MASSV. It was determined* that the default library for the `sin()`, `cos()`, and `exp()` functions on BG/L is the GNU libm library, which is rather slow. MASS and MASSV are highly optimized libraries for these basic mathematical functions, and MASSV includes vector versions of those functions that can take advantage of improved instruction scheduling and temporal locality. By calling vector functions directly in the code, we witnessed a 30% increase in performance. Other optimizations consisted of loop unrolling and replacing calls to the Fortran `aint(x)` intrinsic function by `real(int(x))`. `aint(x)` results in a function call that is much slower than using the equivalent `real(int(x))`. These combined optimizations resulted in a performance improvement of almost 60% over original runs. It is important to mention that the results presented here are for virtual node mode. GTC has shown an extremely high efficiency of over 95% when using the second core on BG/L nodes, which is quite promising as more cores are added to upcoming processor roadmaps.

Another interesting optimization performed on BGW was processor mapping. The 3D torus used for point-to-point communications is ideally suited for the GTC toroidal geometry. Additionally, the number of toroidal domains used in the GTC simulations exactly match one of the dimensions of the BG/L network torus. Thus by using an explicit mapping file that aligns the main point-to-point communications that occur when particles move from one domain to the next, we were able to improve the performance of the code by 30% over the default mapping.

*The authors thank Bob Walkup for his BG/L optimization insights.

4 ELBM3D: Lattice Bolzmann Fluid Dynamics

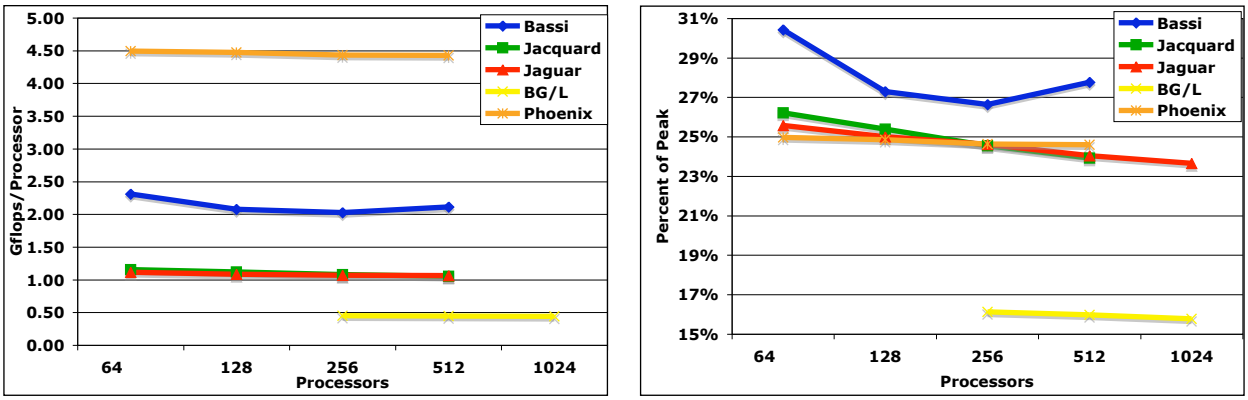
Lattice-Boltzmann methods (LBM) have proved a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids [20]. The basic idea is to develop a simplified kinetic model that incorporates the essential physics, and reproduces correct macroscopic averaged properties.

While LBM methods lend themselves to easy implementation of difficult boundary geometries (e.g. by the use of bounce-back to simulate no slip wall conditions), here we report on 3D simulations under periodic boundary conditions, with the spatial grid and phase space velocity lattice overlaying each other. Each lattice point is associated with a set of mesoscopic variables, whose values are stored in vectors proportional to the number of streaming directions. The lattice is partitioned onto a 3-dimensional Cartesian processor grid, and MPI is used for communication — a snapshot of the communication topology is shown in Figure 1(b), highlighting the relatively sparse communication pattern. As in most simulations of this nature, ghost cells are used to hold copies of the planes of data from neighboring processors. For ELBM3D, a non-linear equation must be solved for each grid-point and at each time-step so that the collision process satisfies certain constraints. Since this equation involves taking the logarithm of each component of the distribution function the whole algorithm becomes heavily constrained by the performance of the `log()` function.

4.1 Experimental results

Strong-scaling results for a system of 512^3 grid points are shown in Figure 3 for both (a) raw performance and (b) percentage of peak. For each of the superscalar machines the code was restructured to take advantage of specialized `log()` functions — ASSV library for IBM and ACML for AMD — that compute values for a vector of arguments. (The benefits of these libraries are discussed in Section 3.) Using this approach gave ELBM3D a performance boost of between 15–30% depending on the architecture. For the X1E, the innermost gridpoint loop was taken inside the non-linear equation solver to allow for full vectorization. After these optimizations, ELBM3D has a kernel of fairly high computational intensity and a percentage of peak of 15–30% on all architectures.

ELBM3D shows good scaling across all of our evaluated platforms. This is due to a lack of load balance issues, and only nearest neighbor point-to-point messaging being required. As expected, the parallel overhead increases as the ratio of communication to computation increases. The parallel efficiency on going to higher concurrencies shows the



(a) (b)

Figure 3. ELBM3D strong-scaling performance using a 512^3 grid by (a) Gflops/processor and (b) percentage of peak. ALL BG/L data collected on the ANL BG/L system in coprocessor mode.

least degradation on the BG/L system (although the memory requirements of the application and MPI implementation prevents running this size on fewer than 256 processors). Both Phoenix and Jaguar are very close behind, followed by Jacquard and Bassi.

Our experiments bear out the fact that the higher computational cost of the entropic algorithm, as compared to traditional LBM approaches, can be cast in a way that leads to efficient computation on commodity processors. We are thus optimistic that ELBM3D will be able to deliver exceptional performance on planned petascale platforms.

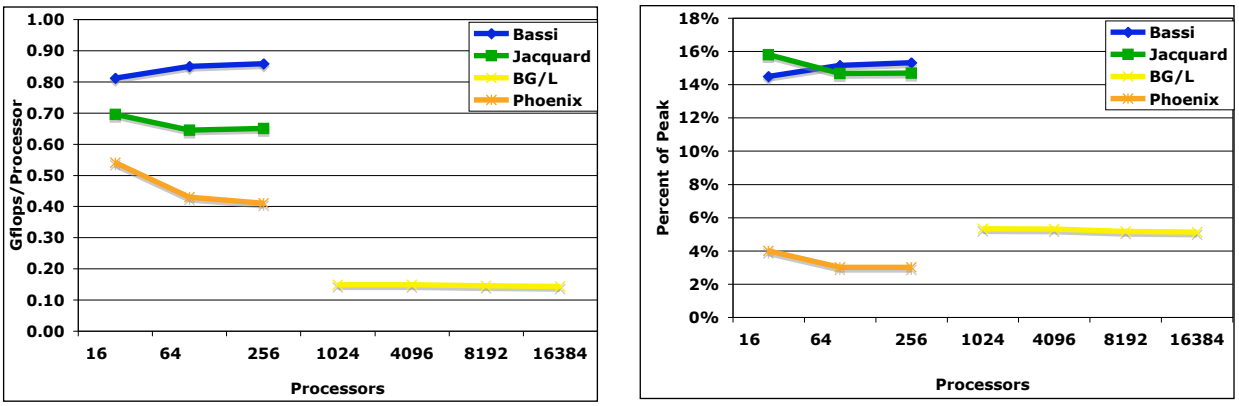
5 Cactus: General Relativity Astrophysics

One of the most challenging problems in astrophysics is the numerical solution of Einstein's equations following from the Theory of General Relativity (GR): a set of coupled nonlinear hyperbolic and elliptic equations containing thousands of terms when fully expanded. The BSSN-MoL application makes use of the Cactus Computational ToolKit [2, 8] to evolve Einstein's equations stably in 3D on supercomputers to simulate astrophysical phenomena with high gravitational fluxes — such as the collision of two black holes and the gravitational waves radiating from that event. The GR components of Cactus use the ADM-BSSN formulation [1] to express Einstein's equations as a system of partial differential equations (four constraint equations and 12 evolution equations) that are evolved forward as an initial value problem. The Method of Lines (MoL) is used to solve the partial differential equations by discretizing in all but one dimension, and then integrating the semi-discrete problem as a system of ODEs in order to improve computational efficiency. For parallel computation, the grid is block domain decomposed so that each processor has a section of the global grid. The standard MPI driver (PUGH) for Cactus solves the PDE on a local grid section and then updates the values at the ghost zones by exchanging data on the faces of its six topological neighbors — resulting in the regular communication topology graph shown in Figure 1(c).

5.1 Experimental results

Figure 4 presents weak-scaling performance results for the Cactus BSSN-MoL application using an 60^3 per processor grid. In terms of raw performance, the Power5-based Bassi clearly outperforms any other systems, especially the BG/L where the Gflops/P rate and the percentage of peak performance is somewhat disappointing. The lower computational efficiency of BG/L is to be expected from the simpler dual-issue in-order PPC440 processor. However, while the per-processor performance of BG/L is somewhat limited, the scaling behavior is impressive, achieving near perfect scalability for up to 16K processors. This is (by far) the largest Cactus scaling experiment to date, and shows extremely promising results. Due to memory constraints we could not conduct virtual node mode simulations for the 60^3 data set, however further testing with a smaller 50^3 grid shows no performance degradation for up to 32K (virtual node) processors. This strongly suggests that the Cactus application will scale to much larger, petascale, systems. Additional investigations with processor topology mappings on the BG/L showed no significant effects.

The Jacquard cluster shows modest scaling, which is probably due to the (relatively) more loosely coupled nature of this system as opposed to the tight software and hardware interconnect integration of the other platforms in our study. Bassi shows excellent scaling but the size of the largest concurrency was significantly smaller compared to that of the BG/L so it remains to be seen if IBM's Federation HPS interconnect will scale to extremely larger systems.



(a) (b)

Figure 4. Cactus weak scaling experiments on a 60^3 per processor grid in (a) Gflops/processor and (b) percentage of peak. All BG/L data was run on BGW. Phoenix data shown on Cray X1 platform.

Phoenix, the Cray X1 platform, showed the lowest computational performance of our evaluated systems. The most costly procedure for the X1 was the computation of radiation boundary condition, which continued to drag performance down despite considerable effort to rewrite it in vectorizable form. In previous studies [3], the vectorized boundary conditions proved beneficial on a number of vector platforms including the NEC SX-8 and Earth; however the X1 continued to suffer disproportionally from small portions of unvectorized code due to the large differential between vector and scalar performance, highlighting that notions of architectural balance cannot focus exclusively on bandwidth (bytes per flop) ratios.

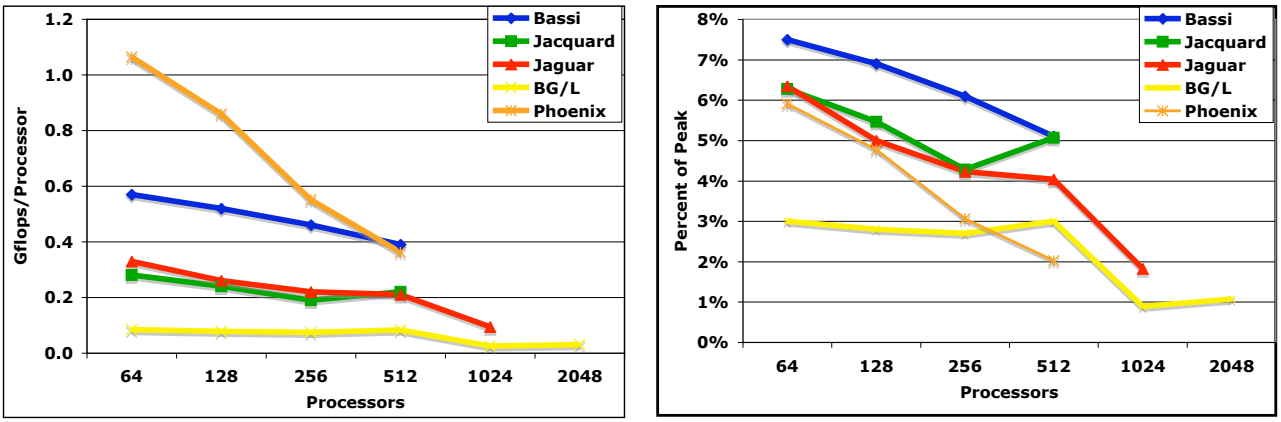
6 BeamBeam3D: High Energy Physics

BeamBeam3D [15] models the colliding process of two counter-rotating charged particle beams moving at close to the speed of light. An accurate modeling of the beam-beam interaction is essential to maximizing the luminosity in high energy accelerator ring colliders.The application performs a 3D particle-in-cell computation that contains multiple models (weak-strong, strong-strong) and multiple collision geometries (head-on, long-range, crossing angle). It tracks macroparticles in colliders using a transfer map.The simulated particles are deposited onto a three-dimensional grid to calculate the 3D charge density distribution. At collision points, the electric and magnetic field are calculated self-consistently by solving the Vlasov-Poisson equation using Hockney's FFT method. Then the electric field and magnetic field are calculated on the grid and reinterpolated back to the macroparticles. The macroparticles are advanced in momentum space using these fields plus external fields from accelerator forces and focusing elements. The parallel implementation utilizes a particle-field decomposition method to achieve load balance. BeamBeam3D's communication is dominated by the expensive global operations to gather the charge density, broadcast the electric and magnetic fields, and perform transposes for the 3D FFTs — this high volume of global message exchange communication can be seen in the topology graph of Figure 1(d).

6.1 Experimental results

For the strong-scaling experiments conducted in this study, we examine a 5 million particle simulation using grid resolutions of 256x256x32; comparative performance data are shown in Figure 5(a). In terms of absolute performance, Phoenix delivers the fastest time-to-solution on 64 processors, almost twice the rate of the next fastest system (Bassi). The multi-streaming vector processor on Phoenix performs excellently on the local computations. However, Phoenix performance degrades quickly with increasing concurrency, and is surpassed by Bassi at 512 processors. The relative degradation on the Phoenix system is partially due to the high communication to computation ratio for BeamBeam3D — at 256 processors over 50% of Phoenix's runtime is spent on communication. Alternative programming paradigms, such as the UPC or CAF global address space languages could potentially improve the Phoenix communication bottleneck [19] compared with the current MPI approach. Additionally, Phoenix performance degrades at high concurrencies due to decreasing vector lengths for this fixed size problem, whereas superscalar platforms generally benefit due to increased cache reuse.

Results also show that Jaguar and Jacquard attain nearly equivalent performance and scalability behavior. Note that although these two platforms use similar Opteron processor technology, they are integrated with vastly different interconnect networks. For this communication-intensive exper-



(a) (b)

Figure 5. BeamBeam3D strong-scaling performance on a $256^2 \times 32$ grid using 5 million particles in (a) Gflops/processor and (b) percentage of peak. For BG/L data, ANL results shown for $P \leq 512$, BGW results for $P = 1024, 2048$.

iment, neither the XT3 nor the InfiniBand approaches confer a performance advantage. However, both of the Opteron systems are almost 1.8x slower than Bassi on 512 processors. Looking at the BG/L system, results show a significant slowdown compared to the other evaluated platforms (almost 4.5x slower than Bassi for P=512).

Figure 5(b) presents BeamBeam3D's sustained percentage of peak for our evaluated systems, showing that no platform attained more than about 5% of theoretical peak. For 512 processors, Bassi achieves the highest rate (5.1%), followed by Jacquard (5%), Jaguar (4%), BG/L (3%) and Phoenix (2%). A number of factors contribute to this low efficiency: indirect data addressing, substantial amounts of global all-to-all communication, and extensive data movement (which does not contribute any flops to the calculation). Due to the large volume of global communication, the parallel efficiency generally declines quickly on all evaluated platforms.

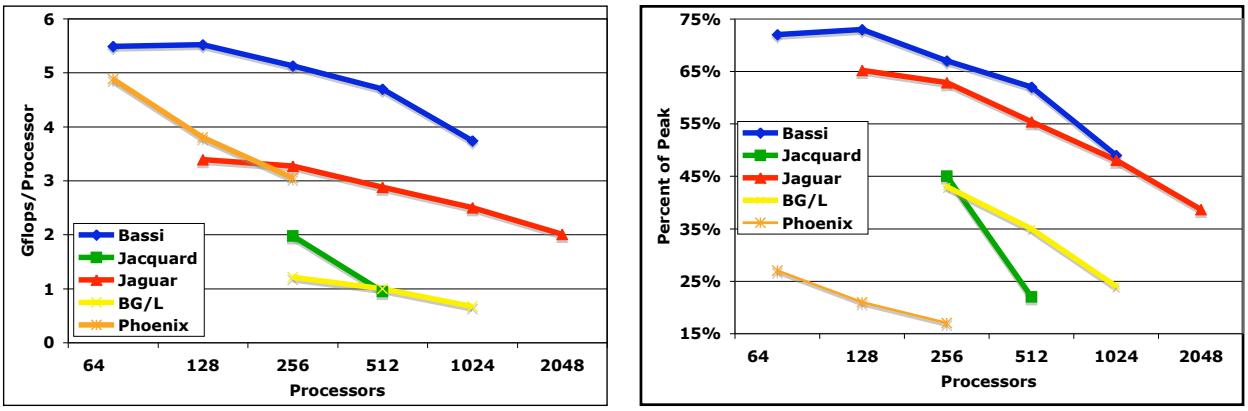
Finally, we highlight the 2048-way BG/L results, representing the highest concurrency BeamBeam3D calculation performed to date. Higher scalability experiments are not possible for this problem size, since there are a limited number of available subdomains. This is because a two-dimensional grid decomposition scheme was chosen to minimize communication requirements. Note that although both BB3D and GTC are PIC codes simulating charged particle interactions, they simulate very different underlying systems. GTC can effectively implement a toroidal grid-based domain decomposition due to the relatively small movements of the particles; however, the particles in BB3D move extensively within each beam, making a pure domain decomposition too expensive in terms of required volume of communication. Nonetheless, we believe that BeamBeam3D has the potential to become a petascale application, but extensive algorithmic reengineering would be required to incorporate additional decomposition schemes and eliminate communication bottlenecks in order to achieve the vast degree of required parallelism.

7 PARATEC: First Principles Materials Science

PARATEC (PARAllel Total Energy Code [14]) performs ab-initio quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. The pseudopotentials are of the standard norm-conserving variety. Forces can be easily calculated and used to relax the atoms into their equilibrium positions. PARATEC uses an all-band conjugate gradient (CG) approach to solve the Kohn-Sham equations of Density Functional Theory (DFT) and obtain the ground-state electron wave functions. Much of the computation time (typically 60%) involves FFTs and BLAS3 routines, which run at a high percentage of peak on most platforms. In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using parallel 3D FFTs to transform the wave functions between the two spaces. The global data transposes within these FFT operations — as seen in Figure 1(e) — account for the bulk of PARATEC's communication overhead, and can quickly become the bottleneck at high concurrencies.

7.1 Experimental results

Figure 6 presents strong-scaling performance results for a 488 atom CdSe (Cadmium Selenide) Quantum Dot (QD) system which has important technological applications due to its photoluminescent properties. Due to the use



(a) (b)

Figure 6. PARATEC strong-scaling performance on a 488 atom CdSe quantum dot. Power5 data for P=1024 was run on the LLNL Purple system†. The BG/L data, collected on the BGW, is for a 432 atom bulk silicon due to memory constraints. Phoenix X1E data was collected using an X1 binary.

of BLAS3 and optimized one dimensional FFT libraries, which are highly cache resident, PARATEC obtains a high percentage of peak on the different platforms studied. The results for BG/L are for a smaller system (432 atom bulk silicon) due to memory constraints.

Results show that the Power5-based Bassi system obtains the highest absolute performance of 5.49 Gflops/P on 64 processors with good scaling to larger processor counts. The fastest Opteron systems (3.39 Gflops/P) was Jaguar (XT3) running on 128 processors. (Jacquard did not have enough memory to run the QD system on 128 processors.) The higher bandwidth for communications on Jaguar (see Table 1) allows it to scale better than Jacquard for this communication-intensive application. The BG/L system has a much lower single processor performance than the other evaluated platforms due to a relatively low peak speed of only 2.8 GF/s. BG/L's percent of peak drops significantly from 512 to 1024 processors, probably due to increased communication overhead when moving from a topologically packed half-plane of 512 processors to a larger configuration. The smaller system being run on the BG/L (432 atom bulk silicon) also limits the scaling to higher processor counts. Overall, Jaguar obtained the maximum aggregate performance of 4.02 Tflops on 2048 processors.

Looking at the vector system, results show that the Phoenix X1E achieved a lower percentage of peak than the other evaluated architectures; although in absolute terms, Phoenix performs rather well due to the high peak speed of the MSP processor*. One reason for this is the relatively slow performance of the X1E scalar unit compared to the vector processor. In consequence, the X1E spends a smaller percentage of the total time in highly optimized 3D FFTs and BLAS3 libraries than on any of the other machines. The other code segments are handwritten F90 routines and have a lower vector operation ratio.

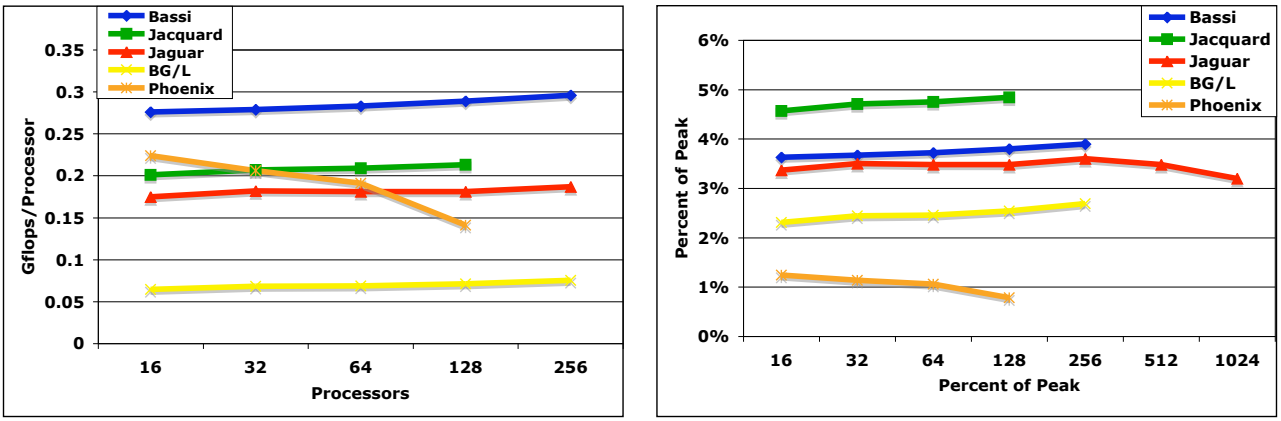
PARATEC results do not show any clear advantage for a torus versus a fat-tree communication network. The main limit to scaling in PARATEC is our handwritten 3D FFTs, where all-to-all communications are performed to transpose the data across the machine. In a single 3D FFT the size of the data packets scales as the inverse of the number of processors squared. PARATEC can perform an all-band calculations, allowing the FFT communications to be blocked, resulting in larger message sizes and avoiding latency problems. Overall, the scaling of the FFTs is limited to a few thousand processors; thus in order to utilize PARATEC at the petascale level with tens (or hundreds) of thousands of processors, we plan to introduce a second level of parallelization over the electronic band indices. This will greatly benefit the scaling and reduce per processor memory requirements on architectures such as BG/L.

8 HyperCLaw: Hyperbolic AMR Gas Dynamics

Adaptive mesh refinement (AMR) is a powerful technique that reduces the computational and memory resources required to solve otherwise intractable problems in computational science. The AMR strategy solves the system of partial differential equations (PDEs) on a relatively coarse grid, and dynamically refines it in regions of scientific interest or where the coarse grid error is too high for proper numerical resolution. HyperCLaw is a hybrid C++/

†Purple, located at LLNL, is architecturally similar to Bassi and contains 12,208 IBM Power5 processors. The authors thank Tom Spelce and Bronis de Supinksi of LLNL for conducting the Purple experiments.

*Results on the X1E were obtained by running the binary compiled on the X1, as running with an optimized X1E generated binary (-O3) caused the code to freeze. Cray engineers are investigating the problem.



(a) (b)

Figure 7. HyperCLaw weak-scaling performance on a base computational grid of 512x64x32 in (a) Gflops/processor and (b) percentage of peak. All BG/L data collected on ANL system.

Fortran AMR code developed and maintained by CCSE at LBNL [7, 16] where it is frequently used to solve systems of hyperbolic conservation laws using a higher-order Godunov method. The HyperCLaw code consists of an applications layer containing the physics classes defined in terms of virtual functions. The basic idea is that data blocks are managed in C++ in which ghost cells are filled and temporary storage is dynamically allocated so that when the calls to the physics algorithms (usually finite difference methods implemented in `Fortran`) are made, the same stencil can be used for all points and no special treatments are required.

8.1 Experimental results

The HyperCLaw problem examined in this work profiles a hyperbolic shock-tube calculation, where we model the interaction of a Mach 1.25 shock in air hitting a spherical bubble of helium. This case is analogous to one of the experiments described by Haas and Sturtevant [10]. The difference between the density of the helium and the surrounding air causes the shock to accelerate into and then dramatically deform the bubble. The base computational grids for the problems studied is $512 \times 64 \times 32$. These grids were adaptively refined by an initial factor of 2 and then a further factor of 4, leading to an effective resolution of $4096 \times 512 \times 256$.

Figure 1(f) shows the interprocessor communication topology of the HyperCLaw calculation. Note that each processor has a surprisingly large number of communicating partners, making the topology graph look more like a many-to-many pattern rather than a simple nearest neighbor algorithm. This highlights the added complexity of maintaining hierarchical grids throughout the calculation. Understanding the evolving communication requirements of AMR simulations will be the focus of future work.

Figure 7 presents the absolute runtime and percentage of peak for the weak-scaling HyperCLaw experiments. In terms of absolute runtime (at P=128), Bassi achieves the highest performance followed by Jacquard, Jaguar, Phoenix, and finally BG/L (the Phoenix and Jacquard experiments crash at P$\geq$256; system consultants are investigating the problems). Observe that all of the platforms achieve a low percentage of peak; for example at 128 processors, Jacquard, Bassi, Jaguar, BG/L, and Phoenix achieve 4.8%, 3.8%, 3.5%, 2.5%, and 0.8% respectively. Achieving peak performance on the BG/L requires both (double hummer) FPUs to be saturated with work; since it is very difficult for the compiler to effectively generate these types of instructions, BG/L peak performance is most likely to be only half of the stated peak. With this in mind, the BG/L would achieve a sustained performance of around 5%, commensurate with the other platforms in our study. Note that although these are weak-scaling experiments in the numbers of grids, the volume of work increases with higher concurrencies due to increased volume of computation along the communication boundaries; thus, the percentage of peak generally increases with processor count.

Although Phoenix performs relatively poorly for this application, especially in terms of its attained percentage of peak, it is important to point out that two effective X1E optimization were undertaken since our initial study into AMR vector performance [22]. Our preliminary study showed that the *knapsack* and *regridding* phases of HyperCLaw were largely to blame for limited X1E scalability, cumulatively consuming almost 60% of the runtime for large concurrency experiments. The original knapsack algorithm — responsible for allocating boxes of work equitably across the processors — suffered from a memory inefficiency. The updated version copies pointers to box lists during the swapping phase (instead of copying the lists themselves), and

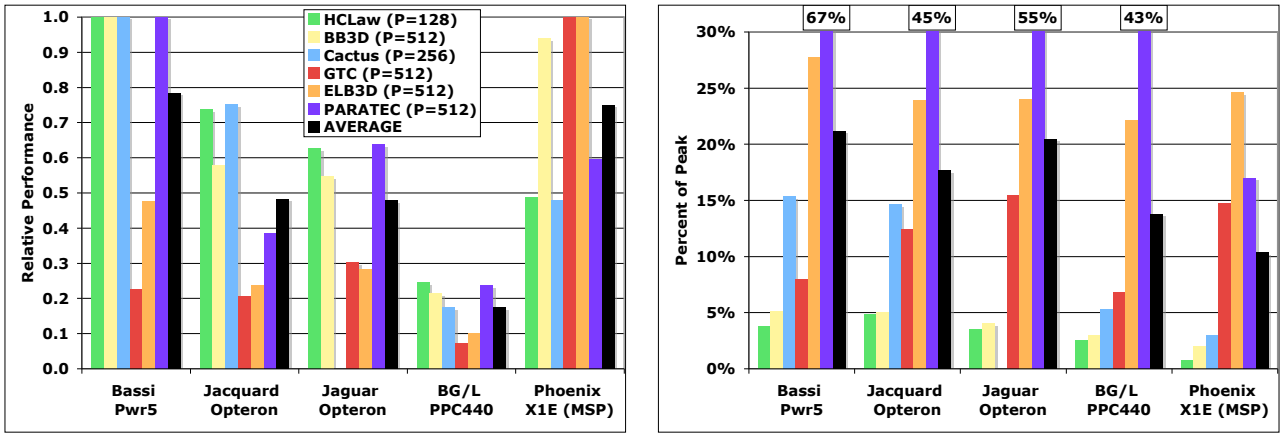


Figure 8. Summary of results for largest comparable concurrencies (a) relative runtime performance normalized to fastest system and (b) sustained percentage of peak. Cactus Phoenix results are on the X1 system. BG/L results are shown for P=1024 on Cactus and GTC.

results in knapsack performance on Phoenix that is almost cost-free, even on hundreds of thousands of boxes.

The function of the regrid algorithm is to replace an existing grid hierarchy with a new hierarchy in order to maintain numerical accuracy, as important solution features develop and move through the computational domain. This process includes tagging coarse cells for refinement and buffering them to ensure that neighboring cells are also refined. The regridding phase requires the computations of box list intersection, which was originally implemented in a $\mathcal{O}(N^2)$ straightforward fashion. The updated version utilizes a hashing scheme based on the position in space of the bottom corners of the boxes, resulting in a vastly-improved $\mathcal{O}(NlogN)$ algorithm. This significantly reduced the cost of the regrid algorithm on Phoenix, resulting in improved performance. Nonetheless, Phoenix performance still remains low due to the non-vectorizable and short-vector-length operations necessary to maintain and regrid the hierarchical data structures.

Overall, our HyperCLaw results highlight the relatively low efficiency of the AMR approach. This is due (in part) to the irregular nature of the AMR components necessary to maintain and regrid the hierarchical meshes, combined with complex communication requirements. Additionally, the numerical Godunov solver, although computationally intensive, requires substantial data movement that can degrade cache reuse. Nevertheless, the efficiency gains associated with AMR and high-resolution discretizations more than compensate for the low sustained rate of execution. Other key result of our study are the knapsack and regridding optimizations, which significantly improved HyperCLaw scalability [22]. These improvements in scaling behavior suggest that, in spite of the low efficiency, the AMR methodology is a suitable candidate for petascale systems.

9 Summary and Conclusions

The purpose of any HEC system is to run full-scale scientific codes, and performance on such applications is the final arbiter of a platform's utility; comparative performance evaluation data must therefore be readily available to the HEC community at large. However, evaluating large-scale scientific applications using realistic problem sizes on leading supercomputing platforms is an extremely complex process, requiring coordination of application scientists in highly disparate areas. Our work presents one of the most extensive comparative performance results on modern supercomputers available in the literature.

Figure 8 shows a summary of results using the largest comparable concurrencies for all six studied applications and five state-of-the-art parallel platforms, in relative performance (normalized to the fastest system) and percentage of peak. Results show that the Power5-based Bassi system achieves the highest raw performance for four of our six applications, thanks to dramatically improved memory bandwidth (compared to its predecessors), and increased attention to latency hiding through advanced prefetch features. The Phoenix system achieved impressive raw performance on GTC and ELBM3D; however, application with nonvectorizable portions suffer greatly on this architecture due the imbalance between the scalar and vector processors. Comparing the two Opteron systems, Jacquard and Jaguar, we see that, in general, sustained performance is similar between the two platforms. However, for some applications such as GTC and PARATEC, the tight integration of Jaguar's XT3 interconnect results in significantly better scalability at high concurrency compared with Jacquard's commodity-based InfiniBand network. The BG/L platform attained the lowest raw and sustained performance on our

suite of applications; however, results at very high concurrencies show impressive scalability characteristics and potential for attaining petascale performance.

Results also indicate that our evaluated codes have the potential to effectively utilize petascale resources. However, some applications, such as PARATEC and Beam-Beam3D, will require significant reengineering to incorporate the additional levels of parallelism necessary to utilize vast numbers of processors. Other applications, including the lattice-Bolzmann ELBM3D and the dynamically adapting HyperCLaw simulation, are already showing scaling behavior with promising prospects to achieve ultra-scale. Finally, two of our tested codes, Cactus and GTC, have successfully demonstrated impressive scalability up to 32K processors on the BGW system. A full GTC production simulation was also performed on 32,768 processors and showed a perfect load balance from beginning to end. This, combined with its high efficiency on multi-core processors, clearly qualifies GTC as a primary candidate to effectively utilize petascale resources.

Overall, these extensive performance evaluations are an important step toward conducting simulations at the petascale level, by providing computational scientists and system designers with critical information on how well numerical methods perform across state-of-the-art parallel systems. Future work will explore a wider set of computational methods, with a focus on irregular and unstructured algorithms, while investigating a broader set of HEC platforms, including the latest generation of multi-core technologies.

Acknowledgments

The authors would like to gratefully thank Bob Walkup for optimizing GTC on the BG/L as well as Tom Spelce and Bronis de Supinksi of LLNL for conducting the Purple experiments. The authors also thank IBM Watson Research Center for allowing BG/L access via the BGW Consortium Day. All authors from LBNL were supported by the Office of Advanced Scientific Computing Research in the Department of Energy Office of Science under contract number DE-AC02-05CH11231. Dr. Ethier was supported by the Department of Energy under contract number DEAC020-76-CH-03073 and by the GPSC SciDAC project.

References

[1] M. Alcubierre, G. Allen, B. Brügmann, et al. Towards an understanding of the stability properties of the 3+1 evolution equations in general relativity. *Phys. Rev. D*, (gr-qc/9908079), 2000.

[2] Cactus Code Server. `http://www.cactuscode.org`.

[3] J. Carter, L. Oliker, and J. Shalf. Performance evaluation of scientific applications on modern parallel vector systems. In *VECPAR: High Performance Computing for Computational Science*, Rio de Janeiro, Brazil, July 10-12, 2006.

[4] T. H. Dunigan Jr., J. S. Vetter, J. B. White III, and P. H. Worley. Performance evaluation of the Cray X1 distributed shared-memory architecture. *IEEE Micro*, 25(1):30–40, January/February 2005.

[5] ORNL Cray X1 Evaluation. `http://www.csm.ornl.gov/~dunigan/cray`.

[6] S. Ethier, W. Tang, and Z. Lin. Gyrokinetic particle-in-cell simulations of plasma microturbulence on advanced computing platforms. *J. Phys. : Conf. Series*, 16, 2005.

[7] C. for Computational Sciences and E. L. B. N. Laboratory. `http://seesar.lbl.gov/CCSE`.

[8] T. Goodale, G. Allen, G. Lanfermann, et al. The Cactus framework and toolkit: Design and applications. In *VECPAR: 5th International Conference, Lecture Notes in Computer Science*, Berlin, 2003. Springer.

[9] F. Gygi, E. W. Draeger, B. R. de Supinski, et al. Large-scale first-principles molecular dynamics simulations on the BlueGene/L platform using the Qbox code. In *Proc. SC2005*, Seattle, WA, Nov 12-18, 2005.

[10] J.-F. Haas and B. Sturtevant. Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities. *Journal of Fluid Mechanics*, 181:41–76, 1987.

[11] HPC challenge benchmark. `http://icl.cs.utk.edu/hpcc/index.html`.

[12] Z. Lin, T. S. Hahm, W. W. Lee, et al. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, Sep 1998.

[13] L. Oliker, J. Carter, M. Wehner, et al. Leading computational methods on scalar and vector HEC platforms. In *Proc. SC2005*, Seattle, WA, Nov 12-18, 2005.

[14] PARAllel Total Energy Code. `http://www.nersc.gov/projects/paratec`.

[15] J. Qiang, M. Furman, and R. Ryne. A parallel particle-in-cell model for beam-beam interactions in high energy ring colliders. *J. Comp. Phys.*, 198, 2004.

[16] C. A. Rendleman, V. E. Beckner, M. L., W. Y. Crutchfield, et al. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3(3):147–157, 2000.

[17] SciDAC: Scientific Discovery through Advanced Computing. `http://www.scidac.gov/`.

[18] STREAM: Sustainable memory bandwidth in high performance computers. `http://www.cs.virginia.edu/stream`.

[19] E. Strohmaier and H. Shan. Apex-Map: A global data access benchmark to analyze HPC systems and parallel programming paradigms. In *Proc. SC2005*, Seattle, WA, Nov 12-18, 2005.

[20] S. Succi. The lattice Boltzmann equation for fluids and beyond. *Oxford Science Publ.*, 2001.

[21] J. Vetter, S. Alam, T. Dunigan, Jr., et al. Early evaluation of the Cray XT3. In *Proc. IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece, April 25-29, 2006.

[22] M. Welcome, C. Rendleman, L. Oliker, et al. Performance characteristics of an adaptive mesh refinement calculation on scalar and vector platforms. In *CF '06: Proceedings of the 3rd conference on Computing Frontiers*, May 2-5,2006.